Detección y seguimiento de palmas y puntas de los dedos en tiempo real basado en imágenes de profundidad para aplicaciones interactivas

Jonathan Robin Langford-Cervantes, Moises Alencastre-Miranda, Lourdes Munoz-Gomez, Octavio Navarro-Hinojosa, Gilberto Echeverria-Furio, Cristina Manrique-Juan, Mario Maqueo

Tecnologico de Monterrey, Campus Santa Fe, Ciudad de México, México

malencastre@itesm.mx

Resumen. Este artículo presenta un método para detectar las puntas de los dedos y palmas de la mano a partir de imágenes de profundidad. La detección de las puntas de los dedos y la palma se hace por medio de análisis morfológico de la región de la mano extraída, es decir, mediante la transformada de distancia y un análisis de un esqueleto inscrito en una imagen de profundidad. El método propuesto está pensado para usarse en aplicaciones interactivas, como videojuegos, y aplicaciones de Interacción Humano-Computadora. Desarrollamos un videojuego de escultura digital 3D que utiliza el método propuesto.

Palabras clave: Seguimiento de manos, clustering, análisis morfológico, aplicaciones interactivas.

Real-Time Palm and Fingertip Tracking Based on Depth Images for Interactive Applications

Abstract. This paper presents a method to detect the fingertips and palm from a human hand from depth images. The palm and fingertip detection is performed via morphological analysis of the extracted hand region, that is, distance transform and an analysis of an inscribed skeleton in the depth image. The proposed method is intended to be used in interactive applications, such as video games, or Human Computer Interaction applications. We developed a 3D digital sculpting video game that uses the proposed method.

Keywords: Hand tracking, point clustering, morphological analysis, interactive applications.

1. Introducción

Recientemente ha habido mucho énfasis en la investigación enfocada a la interacción humano-computadora (HCI, por Human-Computer Interaction en

inglés), proponiendo crear interfaces que utilizan directamente las habilidades naturales de comunicación y manipulación de los humanos. De las diferentes partes del cuerpo, las manos son las que son más efectivas para ese objetivo, debido a su destreza para comunicación y manipulación.

Previamente, se ha utilizado hardware especializado para realizar dicha tarea (e.g. sensores ópticos, guantes de datos). Aunque con ellos se pueden obtener mediciones precisas en tiempo real, son incómodos y de costos elevados. Por esto, los métodos basados en visión por computadora han sido la corriente principal en este campo, ya que son más baratos y permiten una experiencia de interacción más natural.

En interfaces basadas en visión por computadora, comúnmente se utiliza detección y seguimiento de manos para permitir al usuario interacciones como son el control del cursor, navegación en 3D, y reconocimiento de gestos dinámicos. Sin embargo, realizar el seguimiento de las manos correctamente es un reto, ya que los muchos grados de libertad en ellas implican una gran variabilidad en su apariencia, además de auto-oclusiones. Con la llegada de las cámaras de luz estructurada (también llamadas cámaras o sensores de profundidad, o RGB-D) y de tiempo de vuelo (time-of-flight), se ha vuelto disponible la información de profundidad en una imagen, lo que ha traído métodos pioneros para capturar la interacción del usuario.

Aunque hay muchas soluciones para seguimiento de manos basadas en cámaras de profundidad, presentan algunas limitaciones. Algunos enfoques dependen de métodos lentos de optimización, procesamiento en paralelo con tarjetas de video programables, o instalaciones inconvenientes [26]. Además, la mayoría no son adecuadas para el uso en aplicaciones interactivas, como videojuegos, debido a su complejidad computacional.

En el presente trabajo, se tiene como objetivo el desarrollo de un método que sea capaz de detectar las palmas y puntas de los dedos de una mano en base a la información obtenida con un sensor de profundidad. Se busca que la detección sea en tiempo real, con el propósito de que se pueda usar en aplicaciones interactivas, como son los videojuegos.

2. Trabajo relacionado

El seguimiento de palmas y puntas de dedos suele llevarse a cabo con métodos basados en visión monocular. Sin embargo, dada la flexibilidad de las manos, las auto-oclusiones, el color de la piel, y las condiciones de iluminación, entre otros, continúa siendo una tarea desafiante. Múltiples enfoques basados en visión para estimación de poses de manos y dedos han propuesto la integración de cámaras de profundidad, ya que son invariantes a la luz y al color de la piel.

Los métodos basados en modelos [12,17–19] usan un modelo virtual de mano que consiste en articulaciones cinemáticas, representando sus grados de libertad, para comparar con observaciones reales, y ajustarlas a las del modelo. En [17] se usan 8 cámaras sincronizadas. En cada una, mapas de color de piel y de bordes forman indicadores de la presencia de una mano, éstos se combinan y se

comparan con un modelo predefinido para deducir su pose. Esta implementación resuelve eficientemente un problema de seguimiento de manos con grados de libertad completos y oclusiones con el método de Particle Swarm Optimization, que puede paralelizarse eficientemente [25]. En GPU, se ejecuta en promedio a 15 cuadros por segundo, es sensible a cambios de iluminación, y requiere una instalación laboriosa.

Otro método basado en modelo se presenta en [26], utiliza tanto información de marcadores de un sistema óptico de captura de movimiento, como los datos RGB-D de un Kinect. Así, se adquiere un rango amplio y de alta fidelidad de datos de movimiento de las manos. Los dos dispositivos son complementarios entre sí, ya que se enfocan en diferentes aspectos de las acciones de las manos. Los sistemas basados en marcadores obtienen datos de posición 3D de alta resolución a gran velocidad, pero suelen no ser capaces de reconstruir precisamente las articulaciones de la mano en 3D, particularmente si hay auto-oclusiones. Complementado con la imagen RGB-D, el efecto se reduce significativamente, y se puede reconstruir la información del movimiento con alta calidad.

Si no se necesita detectar los grados de libertad de la mano, se puede utilizar un enfoque basado en características [6, 10, 13, 21] en el que sólo algunas, como las puntas de los dedos y el centro de la palma, o gestos simples, son detectados. Raheja et al. [21] utilizan un histograma de intensidad para detectar la dirección de la mano, el final de la muñeca, y el final de los dedos. Este método es rápido y eficiente para manos y puntas, pero sólo es útil cuando todos los dedos están presentes y abiertos completamente. Para tareas como señalar, cuando las puntas de los dedos son los elementos más importantes, Hongyong et al. [10] combinan información de color y profundidad para segmentar la mano y encontrar las puntas con detección de bordes. Los contornos aislados se tratan como manchas que se rastrean utilizando el algoritmo de clasificación KNN (K-Nearest Neighbors, o K-vecinos más cercanos), y después se interpretan como gestos.

Du et al. [6] segmentan la mano mediante umbralización de los datos de profundidad. Después, localizan los puntos convexos y cóncavos para predecir los dedos. Este método es capaz de detectar dedos en tiempo real con una precisión del 94 % para una mano. El trabajo presentado por Li et al. [13] también utiliza detección de convexidad, pero puede rastrear dos manos aplicando el algoritmo de K-Means después del umbral de profundidad. También puede detectar más gestos aplicando tres capas de clasificadores, con una precisión de 84 % para una mano, aumentando a más de 90 % si las dos hacen el mismo gesto. Liang et al. [14] obtuvieron errores de 0.69cm a 2.51cm en detección de dedos con un algoritmo que encuentra las palmas, y después restringe el movimiento posible a detectar, utiliza un rectángulo local, y puntos de cámino geodésico más corto para detectar las puntas de los dedos. Estos trabajos se enfocaron en obtener un alto porcentaje de precisión, pero no se ejecutan en tiempo real.

Suau et al. [23] lograron desempeño en tiempo real con una imagen de profundidad de baja resolución, siguiendo la cabeza y las manos relativas a la posición de ésta. Su desempeño en este aspecto fue mejor que en el SDK

del Kinect 1.0. El estado de la mano (abierto o cerrado) es detectado basado en el área delimitada por el conjunto de puntos. Lograron un error de menos de 10cm. Krejov et al. [11] usaron un detector Viola-Jones para encontrar la cara en una imagen de color, y con su posición establecieron un umbral para detección de manos. Después usaron el algoritmo de Dijkstra para recorrer un grafo ponderado construido a partir de los datos de profundidad del Kinect, para obtener 7 candidatos a puntas de dedo, y descartar falsos positivos como la muñeca. Sus resultados fueron un 80 % de detecciones entre 5.1 y 5.7mm de la posición real.

Hay aplicaciones como [5,9], en las que el problema de detección está resuelto, pero el método utilizado se explica sólo parcialmente. El segundo trabajo utiliza un Kinect para distinguir manos y dedos en una nube de más de 60,000 puntos a 30 cuadros por segundo. El primer trabajo presenta un seguidor de manos articuladas en tiempo real que reconstruye poses complejas con precisión, utilizando el Kinect del Xbox One. También se puede recuperar de fallos en el seguimiento.

El método propuesto busca obtener las puntas de los dedos y la palma de la mano, sin la necesidad de hardware especializado de gran costo. De igual manera, se busca que el método sea capaz de ejecutarse solamente con CPU, haciendo que sea viable su uso en computadoras portátiles, o de bajo costo. A diferencia de otros trabajos, se busca que el proceso de detección se logré en tiempo real, es decir, al menos poder procesar 30 cuadros por segundo. Se enfocó el trabajo en videojuegos, y aplicaciones interactivas similares, donde el uso de palmas y puntas de los dedos, así como gestos simples, como saludar, son suficientes para permitir control e interactividad. Para evaluar el método y proveer una forma de verificar la usabilidad de las manos en HCI, se desarrolló un videojuego de escultura digital 3D que utiliza el método tanto para modificar un modelo 3D preexistente, como para crear un objeto nuevo con "barro virtual".

3. Desarrollo

El método que se propone toma un mapa de profundidad como entrada desde un sensor Kinect, y trata de determinar la posición de las puntas de los dedos y las palmas de las manos para un sólo usuario. El proceso general es el siguiente:

1. Segmentación de las regiones potenciales de la mano usando el mapa de profundidad. El resto de los puntos del mapa se descartan.
2. Agrupamiento de puntos. Se usa un algoritmo de agrupamiento para juntar todos los puntos del mapa segmentado para posteriormente ser procesados.
3. Estimación de la posición de la palma utilizando la transformada de distancia.
4. Selección de puntas de los dedos. Se crea un grafo, que puede verse como el esqueleto para la mano, a partir de los grupos de puntos. Las ramas del grafo se escogen como candidatos a puntas de los dedos. Se aplican reglas adicionales para seleccionar las puntas finales.

3.1. Obtención del mapa de profundidad y segmentación

El mapa de profundidad debe ser segmentado para mantener solamente las regiones que corresponden a las manos y los antebrazos del usuario. La Figura 1 muestra un mapa de profundidad antes de ser filtrado, que comúnmente contiene elementos como la cabeza, los brazos, u otros objetos en escena. Para procesar solamente los puntos de manos y antebrazos, se creó una caja delimitadora que rodeará únicamente el área de interés, y deja fuera el resto de los puntos de la escena. Un mapa filtrado puede verse en la Figura 2. Escogimos este acercamiento para permitir a los usuarios de una aplicación configurar su espacio de trabajo y no restringir el posicionamiento del Kinect, o la gente usándolo.

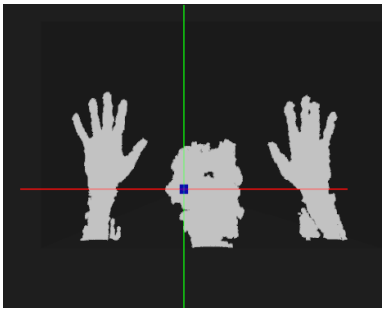

Fig. 1: Nube de puntos 3D antes de filtrarse

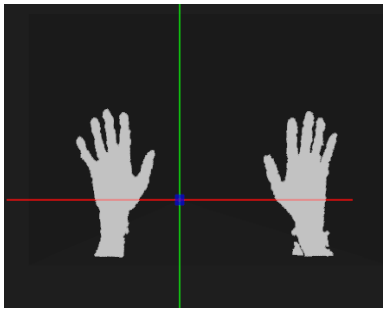

Fig. 2: Nube de puntos 3D después de filtrarse

3.2. Agrupamiento de puntos en grupos

Después de segmentar el mapa de profundidad, se agrupan los puntos 3D en grupos significativos que representan cada mano y antebrazo, para después obtener el grafo que servirá como esqueleto. Se consideró el uso de K-Means [15] y DBSCAN [8], debido a su eficiencia computacional y su efectividad al encontrar grupos significativos.

Se hicieron pruebas primero con el algoritmo K-Means, buscando en el mapa de profundidad filtrado dos grupos; uno para cada mano. El algoritmo de K-Means produjo dos grupos que se podían analizar posteriormente (como se ve en la Figura 3). Sin embargo, debido a que K-Means necesita un número predefinido de grupos a buscar, hubieron algunos casos en los que las manos estaban muy cerca entre sí, y ocasionaron que el algoritmo generara grupos con puntos que pertenecían a una mano asignados a la otra (ver Figura 4).

DBSCAN trabaja considerando la densidad de los grupos, y produce un número de grupos depende de los datos en sí, resolviendo así los problemas que se tienen con K-Means. Utilizando DBSCAN en un conjunto de datos de prueba, se encontró que los puntos que conforman las manos se agrupaban adecuadamente,

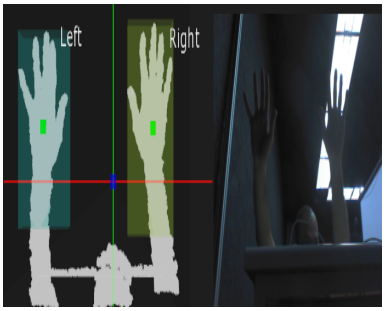

Fig. 3: Grupos generados correctamente usando K-Means

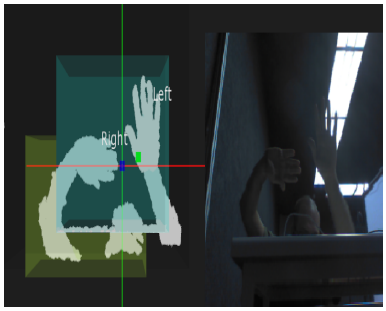

Fig. 4: Asignación incorrecta de puntos a un cluster con K-Means

por lo que se decidió seguir con dicho algoritmo. Sin embargo, el algoritmo no se ejecuta en tiempo real. Por esto, se utilizó una modificación a DBSCAN, propuesta por Navarro et al. [16], que es capaz de procesar nubes de puntos en tiempo real. La modificación a DBSCAN utiliza octrees y un esquema de particionamiento para reducir el tiempo que pasa buscando vecinos. El resultado del agrupamiento de la nube de puntos se puede ver en la Figura 5.

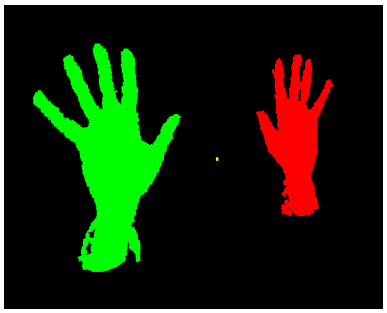

Fig. 5: Grupos finales obtenidos de la nube de puntos

3.3. Estimación de las palmas usando la transformada de distancia

Para estimar el centro de la palma, se utilizó el algoritmo de la transformada de distancia [20]. Para aplicar la transformada y estimar el centro de la palma, se siguieron los siguientes pasos:

- Se proyecta la nube ortogonalmente sobre el eje Z y se crea una imagen binarizada. El resultado se puede ver en la Figura 6 (a).
- Se rellena cualquier hoyo que se haya podido generar en el paso anterior. Mostrado en la Figura 6 (b).

- Se calcula la transformada de distancia de la imagen resultante. Se ve en la Figura 6 (c).
- Se encuentra el punto válido en el mapa de distancia con la distancia más grande. Si hay más de uno, se escoge el que tenga la menor coordenada Y. Ese punto servirá como el centro de la palma. Visto en la Figura 6 (d).

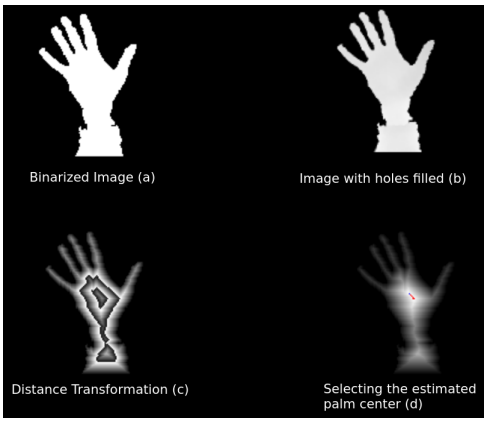

Fig. 6: Estimación del centro de la palma mediante la transformada de distancia en una imagen binarizada

3.4. Selección de puntas de los dedos

Para obtener las puntas de los dedos, se hace un análisis de los grupos obtenidos previamente para estimar la posición de las mismas. Se utilizó una técnica similar a la usada en [22, 24]. La idea principal es generar un esqueleto de curvas centradas de un grupo de puntos 3D, con el propósito de ayudar a estimar la forma de la mano y las puntas de los dedos.

Primero, se genera un octree con los puntos de cada grupo; esto permite utilizarlos en una búsqueda, y procesarlos en tiempo real. Para cada octree, se calcula el centroide de cada uno de sus voxeles, y se aplica DBSCAN para agruparlos a lo largo del eje Y, como se puede ver en la Figura 7. Para cada nuevo grupo que se genera, se calcula su centroide (ver Figura 8), de modo que se convierten en nodos de un esqueleto de curvas.

Para unir cada uno de los nuevos nodos, se diseñaron las siguientes reglas:

- Los nodos se conectarán de abajo hacia arriba en el eje Y.
- Cada nodo se conectará al siguiente conjunto de nodos más cercanos disponible.
- Los nodos buscarán conexiones solamente en los próximos cinco niveles del octree durante la búsqueda. Si el siguiente nodo más cercano está más lejos, no se conectará.

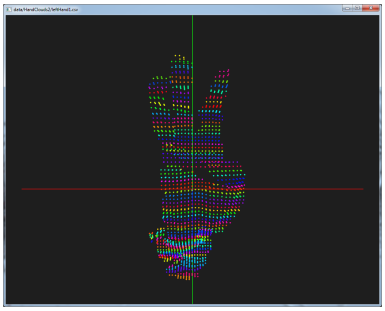

Fig. 7: Puntos del octree agrupados en grupos

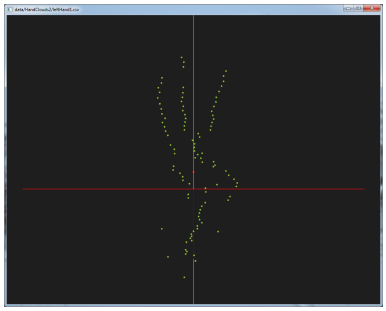

Fig. 8: Centroides de cada capa de los puntos en el octree

Usando esta técnica, se obtiene un grafo que puede verse como el esqueleto de curvas de cada mano. Se busca que los nodos finales del esqueleto sean los candidatos a puntas de los dedos. Sin embargo, el esqueleto generado puede tener múltiples bifurcaciones que deben de ser recortadas ya que también se considerarían como candidatos. Por esto, se eliminaron todos los caminos del esqueleto que tienen menos de dos conexiones subsecuentes.

Los nodos finales de cada camino son los que se considerarán como candidatos a puntas de dedos. Considerando la longitud de cada camino desde la raíz a los candidatos a dedo, todos los caminos se ordenan de forma descendiente, y sólo los primeros cinco se seleccionan. Para validar los candidatos, utilizamos un método geométrico. Definimos un radio de aprobación, que es inversamente proporcional al camino más largo desde el centro de la palma a los candidatos, y todos los que quedan fuera de ese radio se validan como puntas de dedo. En la Figura 9 se puede apreciar la selección de las puntas de los dedos.

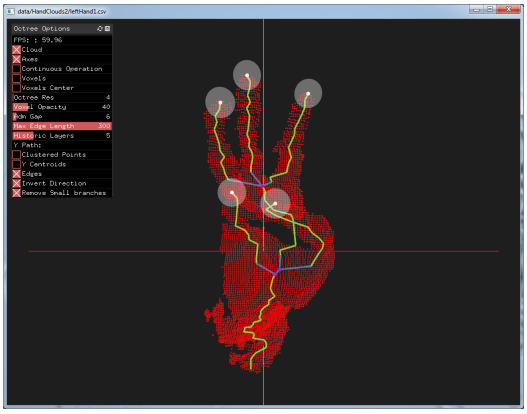

Fig. 9: Selección de puntas de dedos final

4. Resultados

El desarrollo se realizó en C++ utlilizando el SDK v1.8 del Kinect [1]. Se hicieron las siguientes pruebas con un conjunto de datos de muestra, y con una captura en vivo del Kinect. El conjunto de muestra consiste en quince diferentes mapas de profundidad, y cada uno consiste en alrededor de 30,000 puntos 3D, que capturan diferentes posiciones, tamaños, y número de manos. Tanto para el conjunto de muestra como para la captura en vivo, se fijó el Kinect en la misma posición. Se fijó a 80cm en frente del punto de captura, 40cm por debajo de éste, y con un ángulo de elevación de 25 grados.

4.1. Estimación de palmas y puntas de los dedos

La propuesta de solución es capaz de detectar la posición de las puntas de los dedos y las palmas de dos manos a un promedio de 60 cuadros por segundo. En las siguientes Figuras, las puntas de los dedos están marcadas con círculos verdes (pequeños), mientras que las palmas de las manos están marcadas con un círculo rojo (grandes). La Figura 10 muestra una captura en vivo de nuestro sistema, siguiendo palmas y puntas de los dedos de dos manos.

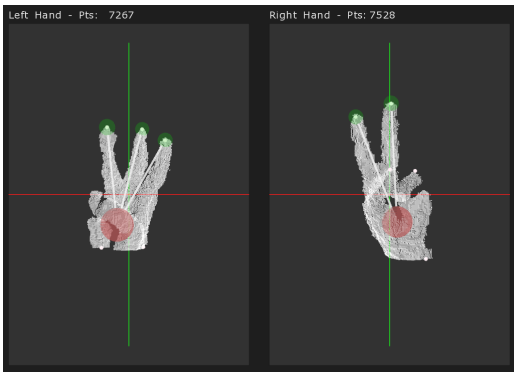

Fig. 10: Captura en vivo de la solución propuesta, siguiendo dos manos.

El sistema puede seguir las palmas y puntas detectadas mientras las manos estén apuntando hacia el Kinect, y las manos no estén ocluidas. El sistema también puede determinar si no hay ninguna punta de dedo presente, como se puede apreciar en la Figura 11 donde sólo el puño se muestra, y sólo la palma se sigue.

El radio de aprobación de la sección 3.4 ayudó con la detección para manos de distintos tamaños. Sin embargo, aunque este nos permitió filtrar algunos candidatos a dedos, también filtra el pulgar cuando está dentro del radio mencionado, como se ve en la Figura 12, donde una mano con 4 dedos apuntando hacia arriba se muestra.

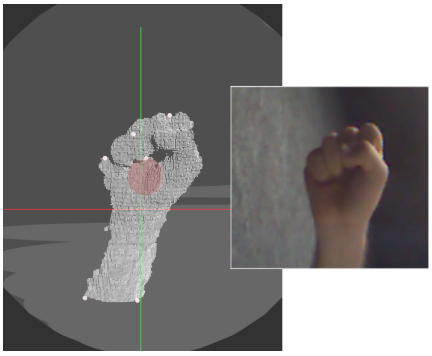

Fig. 11: Sólo la palma es seguida cuando se muestra un puño

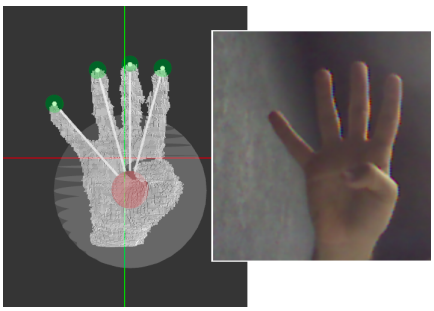

Fig. 12: Sólo 4 puntas de dedo y el centro de la palma se siguen, de una mano mostrando sólo 4 dedos

Para probar la solución más a fondo, se desarrolló un juego simple de escultura digital utilizando el motor de juegos Unity [7]. Dentro del juego, un conjunto de gestos (como agarrar y soltar) fueron programados utilizando la información obtenida con la detección y seguimiento de las palmas y puntas de los dedos. Con esos gestos, pudimos controlar la interfaz de usuario (UI) y los objetos en escena. La UI consistió en botones y barras deslizantes, con los que se interactúa abriendo y cerrando la mano, y arrastrando en el caso de las barras. El juego tiene dos modos: Escultura de barro digital, donde se puede modelar utilizando marching cubes imitando el barro, y escultura digital, donde se usan o deforman formas básicas, como una esfera o un toroide, con el objetivo de modelar algo distinto. Este juego nos mostró que la solución es lo suficientemente precisa como para desarrollar una aplicación sencilla de HCI, que permitió al usuario modelar precisamente diversos objetos. Una muestra del juego se puede ver en las Figuras 13 y 14.

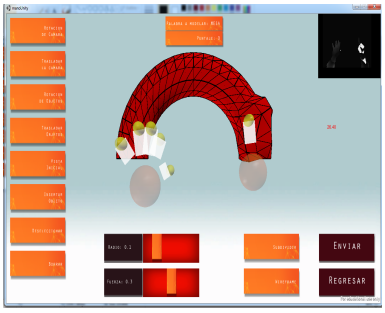

Fig. 13: Juego de ejemplo, esculpiendo un objeto nuevo utilizando un toroide como base

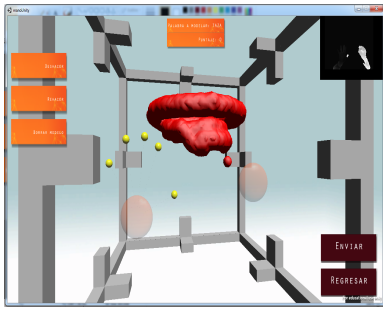

Fig. 14: Juego de ejemplo, modelando un objeto nuevo utilizando marching cubes como barro

5. Conclusiones y trabajo futuro

Se desarrolló un sistema que puede detectar la posición en 3D de las dos palmas y las puntas de los dedos de una mano humana, en tiempo real, utilizando un mapa de profundidad 3D. Aún cuando nuestra solución sólo detecta puntas de los dedos y centros de la palma, y puede perder el pulgar de las manos, produce un resultado aceptable para aplicaciones que necesitan una interfaz simple, y no tienen hardware dedicado, como tarjetas de video, disponible para procesamiento. Algunas de estas aplicaciones pueden ser videojuegos, simuladores, y con la llegada de los sensores de profundidad móviles [2–4], aplicaciones para dispositivos móviles.

Para trabajo a futuro, nos gustaría generar un modelo que detecte los dedos completos en cada mano. Además, la generación del esqueleto de curvas depende de la orientación de la mano (que necesita estar apuntando hacia arriba de frente al sensor de profundidad), por ello nos gustaría implementar un vector de orientación que pueda identificar la orientación de la mano, de modo que el requerimiento previo no sea necesario, generando así una mayor libertad dentro de los movimientos disponibles de las manos.

Referencias

1. Kinect for windows sdk v1.8 (sep 2014), http://www.microsoft.com/en-us/download/details.aspx?id=40278, Accesado: 2014-09-13
2. Atap project tango - google (sep 2014, Accesado: 2014-09-13), https://www.google.com/atap/projecttango/#project
3. Softkinetic - depthsense modules (sep 2014, Accesado: 2014-09-13), http://www.softkinetic.com/products/depthsensemodules.aspx
4. The structure sensor is the first 3d sensor for mobile devices (sep 2014, Accesado: 2014-09-13), http://structure.io/

5. Andrew, F., Daniel, F., Shahram, I., Cem, K., Eyal, K., Ido, L., Christoph, R., Toby, S., Jamie, S., Jonathan, T., Alon, V., Yichen, W.: Fully articulated hand tracking (2014), http://research.microsoft.com/en-us/projects/handpose/default.aspx, Accesado: 2014-10-29
6. Du, H., To, T.: Hand gesture recognition using kinect. Techical Report, Boston University (2011)
7. Engine, U.G.: Unity, http://unity3d.com/, Accesado: 2014-09-13
8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: 2nd International Conference on Knowledge Discovery and Data Mining. vol. 96, pp. 226–231. AAAI Press, Portland, OR, USA (1996)
9. Garratt, G., Leslie, P.K., Russ, T., Tomas, L.P.: Kinect hand detection (2010), http://www.csail.mit.edu/videoarchive/research/hci/kinect-detection, Accesado: 2014-10-29
10. Hongyong, T., Youling, Y.: Finger tracking and gesture recognition with kinect. In: 2012 IEEE 12th International Conference on Computer and Information Technology (CIT). pp. 214–218. IEEE, Chengdu, Sichuan, China (2012)
11. Krejov, P., Bowden, R.: Multi-touchless: Real-time fingertip detection and tracking using geodesic maxima. In: Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on. pp. 1–7. IEEE (2013)
12. Kuznetsova, A., Rosenhahn, B.: Hand pose estimation from a single rgb-d image. In: Advances in Visual Computing, Lecture Notes in Computer Science, vol. 8034, pp. 592–602. Springer Berlin Heidelberg (2013)
13. Li, Y.: Hand gesture recognition using kinect. In: 2012 IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS). pp. 196–199. IEEE, Haiddian District, Beijing, China (2012)
14. Liang, H., Yuan, J., Thalmann, D.: 3d fingertip and palm tracking in depth image sequences. In: Proceedings of the 20th ACM international conference on Multimedia. pp. 785–788. ACM (2012)
15. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297. California, USA (1967)
16. Navarro-Hinojosa, O., Alencastre-Miranda, M.: Dbscan modificado con octrees para agrupar nubes de puntos en tiempo real. In: Memorias del 8º Congreso Mexicano de Inteligencia Artificial. INAOE (2016)
17. Oikonomidis, I., Kyriazis, N., Argyros, A.: Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In: 2011 IEEE International Conference on Computer Vision (ICCV). pp. 2088–2095. IEEE, Barcelona, Spain (Nov 2011)
18. Oikonomidis, I., Kyriazis, N., Argyros, A.A.: Efficient model-based 3d tracking of hand articulations using kinect. In: 22nd British Machine Vision Conference. vol. 1, p. 3. Dundee, United Kingdom (2011)
19. Oikonomidis, I., Kyriazis, N., Argyros, A.A.: Tracking the articulated motion of two strongly interacting hands. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1862–1869. IEEE, Providence, USA (2012)
20. Raheja, J.L., Chaudhary, A., Singal, K.: Tracking of fingertips and centers of palm using kinect. In: 2011 Third International Conference on Computational Intelligence, Modelling and Simulation (CIMSiM). pp. 248–252. IEEE, Langkawi, Malaysia (2011)

21. Raheja, J.L., Das, K., Chaudhary, A.: Fingertip detection: A fast method with natural hand. International Journal of Embedded Systems and Computer Engineering 3(2), pp. 85–89 (2011)
22. Sam, V., Kawata, H., Kanai, T.: A robust and centered curve skeleton extraction from 3d point cloud. Computer-Aided Design and Applications 9(6), pp. 869–879 (2012)
23. Suau, X., Ruiz-Hidalgo, J., Casas, J.R.: Real-time head and hand tracking based on 2.5 d data. Multimedia, IEEE Transactions on 14(3), 575–585 (2012)
24. Tagliasacchi, A., Zhang, H., Cohen-Or, D.: Curve skeleton extraction from incomplete point cloud. In: ACM SIGGRAPH 2009 Papers. SIGGRAPH '09, vol. 28, pp. 71:1–71:9. ACM, ACM, New York, NY, USA (2009)
25. Venter, G., Sobieszczanski-Sobieski, J.: Particle swarm optimization. American Institute of Aeronautics and Astronautics journal 41(8), pp. 1583–1589 (2003)
26. Zhao, W., Chai, J., Xu, Y.Q.: Combining marker-based mocap and rgb-d camera for acquiring high-fidelity hand motion data. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 33–42. Eurographics Association (2012)